Preprint
UCRL-JC-148098

Visualization of Time-Varying Multiresolution Data Using Error-Based Temporal-Spatial Reuse

C. Nuber, E.C. LaMar, B. Hamann, K.I. Joy

This article was submitted to
IEE Visualization 2002, Boston, MA, October 27 – November 1, 2002

April 22, 2002

U.S. Department of Energy
Lawrence
Livermore
National
Laboratory

Visualization of Time-varying Multiresolution Data Using Error-based Temporal-spatial Reuse

Christof Nuber, Eric C. LaMar, Bernd Hamann, and Kenneth I. Joy

Abstract

In this paper, we report results on exploration of two-dimensional (2D) time varying datasets. We extend the notion of multiresolution spatial data approximation of static datasets to spatio-temporal approximation of time-varying datasets. Time-varying datasets typically do not change "uniformly," *i.e.*, some spatial sub-domains can experience only little or no change for extended periods of time. In these sub-domains, we show that approximation error bounds can be met when using sub-domains from other time-steps effectively. We generate a more general approximation scheme where sub-domains may approximate congruent sub-domains from any other time steps. While this incurs an $O(T^2)$ overhead, where T is the total number of time-steps, we show significant reduction in data transmission. We also discuss ideas for improvements to reduce overhead.

1 Introduction

Computer simulations of complex physical phenomena generate extemely high-resolution time-varying datasets. It is desirable to investigate these datasets interactively. Very few of these datasets can be loaded into a computer's main memory. Single time steps are typically loaded. For each new time step, the entire time step must be (re)loaded. For large datasets, even a single time-step can be too large.

This problem can cause an unacceptable stall in applications where real-time behavior is needed, distracting a user. This effect can be particularly distracting in a virtual reality setting, where a rendering budget must be satisfied to guarantee a "smooth" immersive experience.

We avoid reloading of new time steps by amortizing over several rendering cycles. First, in regions of the dataset where there is no change between timesteps, it is not necessary to load that sub-region for a new time-step - the "old" region can be reused without error. Second, We allow a sub-region from one time-step to approximate the same sub-region of a different time step.

This technique is intended to be complementary to compression, differential encoding, or other similar methods used to reduce/compress data size. While the technique discussed in this paper is restricted to 2-d animations, all of the techniques and observations extend to 3-d time-varying data.

The paper is structured as follows: In section 2, we discuss related work. In section 3, we describe our new approach in detail, describing the error tables used, the determination of the error for a region, the management of the regions and the strategy used for selecting regions for replacement. In section 4, we demonstrate how our approach performs when compared to other strategies using two different image sequences.

2 Related Work

Finkelstein et al. [1], were among the first to develop methods for adaptive time-varying animation. They use a multiresolution image technique and a quad-tree to decompose an image. Nodes represent flat regions (*i.e.*, store the average color of the children) and may contain child pointers when higher resolution information is available. The temporal aspect is encoded as a time-spanning binary tree, where one image quad-tree is associated with each node of the binary tree. The root node of the binary tree represents the averaged image from all time steps (*i.e.*, a "motion-blurred" image). The leaf nodes of the binary tree store the imagery from a single time-step. Compression is achieved through first pruning quad-tree nodes when all of the children of a parent node are equal to it within some error tolerance δ. Second, similar sub-trees from different time-steps (frames) are merged into one (*i.e.*, the first instance is saved, all others are deleted, and the pointers originally pointing to these individual sub-trees all point to the first instance). There is no notion of run-time error, only the error associated with the initial construction of the time sequence.

LaMar et al. [2], and Weiler et al. [3], described interactive multiresolution volume visualization systems. The volume is composed into an octree hierarchy of approximations, with each level of the tree half the spatial resolution of the next level. The coarsest approximation is stored at the root node, approximations in the internal nodes, and the original data at the root node. A user-defined importance function, a data error function, and rendering budgets guide the approximation.

LaMar et al. [4] introduced a very fast method for computing error on multiresolution hierarchies for volume rendering applications. The error is computed on the image volume produced by applying a color and opacity transfer function. The key observation is this one: Though the color space and data volume are large, for data sets who's voxels are bytes, there are only 256^2 *unique* pairs of error terms. The frequency of error terms in a node of the octree is recorded in a table is also associated with that node. To evaluate the error associated with a node, only the table must be evaluated. If a volume contains 512^3 (2^{27}) voxels, then the speed-up for computing error for a new transfer function on the root node of the approximation hierarchy over that 512^3 volume, is $2^{27}/2^{16} = 2^{11} = 2048$. Coupled with a lazy evaluation of the error (*i.e.*, error calculated only when a node of the tree is visited), this results in extremely fast error calculation.

Ma et al. [5] used voxel-level quantization, octree encoding, and differential encoding for compression on voxel, spatial, and temporal dimensions, respectively. They claimed a compression of up to 90% for some data sets. Since voxel-level quantization is used, this is a lossy technique and is not appropriate for all visualizations, and certainly not appropriate for archival purposes. They used a technique called "temporal merging," where two or more temporally-consecutive sub-domains are merged. The partial image generated from the first sub-domain of this series is cached and reused for rendering later time-steps. This work has a limited notion of error with respect to time: A sub-region is reloaded when there is any change in data values, and reused if there is no change. Also, since only imagery is cached, re-rendering the volume with a new transfer function or from a new viewpoint requires loading the entire approximation.

Shen et al. [6] introduced the TSP method which decomposes space using an octree, and in each node of the octree, decomposes time using a binary tree. The TSP mechanism can represent, to

a limited degree, error in both spatial and temporal senses. Non-leaf nodes store error terms, not approximations of data. Bricks of original data are stored only at the leaf nodes. Spatial error is used to determine when to approximate a sub-region of a volume with a single color; that is, either a subregion is rendered at original resolution or is approximated by a single color. Temporal error is used to determine whether imagery generated at one time-step can be used to approximate imagery of later time-steps. In both cases, error is estimated using the variance of spatial and temporal subregions. Low variance results in a constant color or reuse of an image; high variance results in rendering original data. Once an image is generated for a subregion, it is cached for possible reuse.

Ellsworth et al. [7] extended the TSP algorithm to hardware texture-based volume rendering. They utilized alternate error metrics, *i.e.*, instead of examining data values they used, the color values resulting from the transfer function are used.

Sutton and Hansen [8, 9] introduced T-BON , which is a time-based extension of the BONO (Branch-On-Need-Octree) [10] method. The BONO method is an octree decomposition of space, where each node stores the extrema values (minimum and maximum) over the corresponding sub-domain. To find a surface for a particular isovalue V, the method starts at the root node and visits all nodes is extrema bracket the isovalue. The T-BON method produces a BONO for each time step. For each new time step, all prior geometry is thrown away, and new geometry is produced, as in the normal BONO method. The per-time-step BONO is accessed in a demand-page fashion. Each node stores the extremal values (minimum and maximum) over the corresponding sub-domain. To find a surface for a particular isovalue V, the method starts at the root node and visits all nodes whose extrema bracket the isovalue.

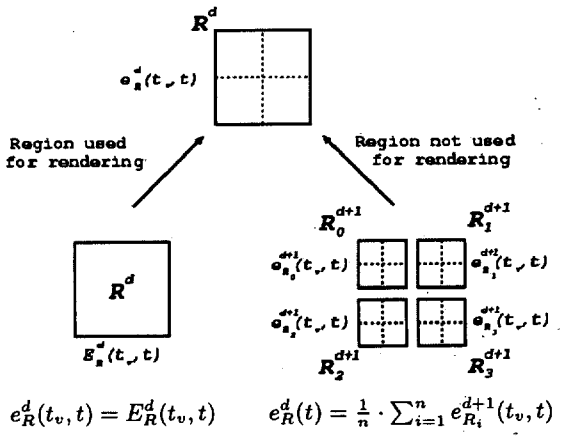
Shen introduced the Temporal Hierarchical Index Tree (THIT) in [11] that constructs a bucketized span space table, where the minimum and maximum values for a cell correspond to the cell's minimum and maximum over *all* time steps. Each table entry contains a temporal subdividing binary tree. The root node records isosurfaces that intersect the cell for all time steps, intermediate nodes record isosurfaces that intersect the cell for intermediate time intervals, and leaf nodes note isosurfaces that intersect the cell for a single time step. The algorithm requires that the grid topology remains fixed over time.

Our algorithm exploits the occurence of small changes and similarities over time in spatial identical regions. Using error tables that describe the difference of a region at a given resolution and the same region at maximum resolution to other time steps, we can determine the exact error introduced by this region. Only when a region does not meet a user-defined error-criterium we need to update this region. This approach allows us to resuse as many regions as possible by exploiting the properties of a multiresolution dataset, where the coarsest representation that meets the error-criterium can be (re)used. Additionally, we are able to provide high-resolution representations of regions with nearly no activity over a long period of time.

3 Approach

Our approach is based on spatial binary subdivision of a time-varying dataset into hierachical organized regions down to a maximum depth d_{max}, together with error tables E_R^d for each region describing the errors over time to determine per region the error introduced with advancing time. During each render-cycle of a timestep t, the visible regions are checked for their error between the visible representation $R^d(t_v)$ and the correct representation $R^{d_{max}}(t)$ where visible means that the region is used for rendering.

Depending on the error of a region it is either reused at timestep t "as is" with the visible representation $R^d(t_v)$, replaced by the representation for the current timestep $R^d(t)$ with $t \neq t_v$ or refined by



$e_R^d(t_v,t) = E_R^d(t_v,t) \qquad e_R^d(t) = \frac{1}{n} \cdot \sum_{i=1}^{n} e_{R_i}^{d+1}(t_v,t)$

Figure 1: Calculation of the error $e_R^d(t_v,t)$ of a region

replacing itself with its subregions R_i^{d+1} to reduce the error displayed. Using this error-driven approach we attempt to reduce the number of subregions reloaded per frame by meeting a given error bound where possible.

In most simulations, changes within a region are small for small timesteps when compared to larger timesteps. Using a maximum error bound e_{max} and exploiting the similarity of regions between consecutive frames out of the visible regions, a set of subregions is determined which must be replaced in order to meet the error requirements. Depending on the current resolution and the change of values over time a region with minimal activity can be reused for several steps before it doesn't meet the given error-criteria anymore ($R^d(t) \approx R^d(t+k), e_R^d(t,t+k) < e_{max}$ for $k > 0$).

The advantage of this approach is that it is independent of the dimension and the representation of the underlying dataset as long as the representation provides multiple resolution and error tables.

3.1 Error Tables

The most accurate way of determining errors $e^d(t_1,t_2)$ between different temporal representations of a region R^d is to use error lookup tables E_R^d. For each region R^d an error table E_R^d is generated which contains the error $e_R^d(t_1,t_2)$ between every timestep t_1 and timestep t_2 of the region at the current depth d in the hierarchy to the corresponding region at the biggest depth d_{max}. Only $e_R^{d_{max}}(t,t) = 0$ can be guaranteed for leaf-regions R^d with $d = d_{max}$ within the hierarchy.

3.2 Error for a Timestep

Each visible region R_v^d manages the time t_v it represents and the error $e_{R_v}^d(t_v,t)$ it introduces. The error $e_{R_v}^d(t_v,t)$ for a region at any given time t is determined either by the value in the error table when the region itself is displayed (e = $E_R^d(t_v,t)$), or by the average sum of the errors of its subregions R_i^{d+1} by $e_R^d(t) = \frac{1}{n} \cdot \sum_{i=1}^{n} e_{R_i}^{d+1}(t_v,t)$ (see Fig. 1).

3.3 Management of Regions

In order to prevent the re-evaluation of errors for all the regions R within the hierarchy during every timestep, the evaluation is restricted to error table lookups for visible regions R_v^d and error recalculation of their parents only. Keeping a sorted list L of all visible regions R_v^d with an error e greater than the given error bound

e_{max} provides us with fast access to those regions that do not meet the specified error condition requiring $e^d_{L(i)} \geq e^{d'}_{L(j)}$ for $i < j$.

3.4 Replacement Strategy

The selection of the regions to be replaced is based upon their current visible error $e^d_{R_v}$ and the maximum number of regions that can be replaced during a single rendering step. As the replacement has no effect on the visible regions (and their error), the errors of all regions as well as the sorted list L must be updated only when time advances to another timestep in the simulation.

Using the list L of regions sorted by their error replacement is done as follows: As long as there are replacable regions in L, we remove the next replacable region R^d. If the error e^d_R is smaller than our error bound e_{max}, we stop the process. If R^d has an actual time stamp ($t_v = t$), then we replace R^d with it's children R^{d+1}_i and add those children to L where $e^{d+1}_{R_i} > e_{max}$ and continue. If R^d_v does not have a current time stamp we calculate the error of the parent R^{d-1} when we replace the current region with the actual representation and keep the representation of the other children. If the calculated parental error e^{d-1}_R is smaller than the error found in the error table E^{d-1}_R we use the parent for the new time step and remove the remaining children from L. If the error in the table $E^d_R(t_v, t)$ of R is smaller than the current error $e^d_R(t_v, t)$ of R, we replace the region with it's actual representation.

This strategy allows us to replace as many tiles as possible during the rendering-step(s) of one simulated timestep by reducing the error with every image rendered until (1) either time advances to another timestep or (2) we reach the given error bound for all tiles visible.

When restricting the maximum traversal-depth of the hierarchy, it is not always possible to reach the given error bound. The errors are computed for the given resolution at the current depth to the maximum depth in the tree. If we restrict the level d of refinement for a region R^d to a level higher than the base-level d_{max}, the region cannot be refined although it may have an error greater than zero. The minimal error possible for such a region, where $d < d_{max}$, is $e^d_R(t_v, t_v)$ with $e^d_R(t_v, t_v) \geq 0$. Depending on the value for e_{max} the term $e^d_R(t_v, t_v) \leq e_{max}$ will not always be true.

4 Results

We have applied our approach to 2D image sequences. For error analysis, we have used a simple mean-square-error between regions, using the grayscale value of the image at different resolutions for difference computations.

4.1 Standard Approaches used for Comparison

4.1.1 Replacement Based on a Fixed Resolution

The simplest approach is to replace each region of the dataset at every timestep at a given resolution. This approach has a fixed and well-known number of replacements and a well-known error for every timestep, which is the error between the original resolution and the selected multiresolution representation of the dataset. This approach has no capability to adapt to a given error-bound, unless we increase the resolution and thus the number of regions to be replaced. We compare our approach to this one with resolutions for the highest and second-highest resolution.

4.1.2 Replacement Based on Adaptive Refinement

A more advanced approach is to use the error tables and reload the regions from scratch for every timestep, starting at the lowest resolution and refining only where the given error-bound requires refinement, *i.e.*, adaptive refinement. This approach adapts to a given error-bound and optimizes the number of regions to be replaced to render the complete region. With the error-method used, regions with only small sub-regions of change when compared to the region the error is calculated for (especially at lower resolutions) are not refined.

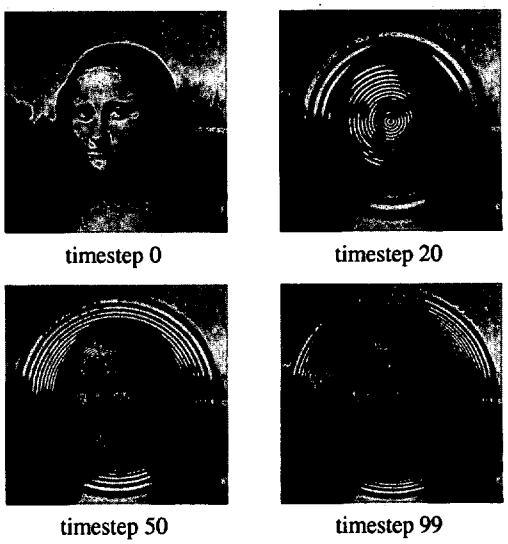


Figure 2: Images from the Mona Lisa sequence

4.2 Datasets Used

For evaluation-purposes we use the following two datasets: an image of the "Mona Lisa" disturbed continuously over 100 timesteps, and an image sequence of the Richtmyer-Meshkov instability simulation, see [12], with 136 images. Both datasets were subdivided into 256 regions at the highest resolution.

4.2.1 Mona Lisa

For simple evaluation, we used a snapshot of the Mona Lisa and disturbed the image I with $I'(x, y) = I(x, y) \cdot |\ (\cos(2^{1-r}) \cdot \Pi\ |$, where r is defined as

$$r = \begin{cases} 1 & \text{if } r > 1 \\ \sqrt{dx^2 + dy^2} & r \leq 1 \end{cases}$$

with $dx = (x - width/2)/(width/2)$ and $dy = (y - height/2)/(height/2)$.

The image sequence consists of 100 images, see Figure 2. Changes within the image occur in the center and propagate to the outer regions of the image in circles with the number of circles increasing by one circle per frame.

4.2.2 Richtmyer-Meshkov Instability

The Richtmyer-Meshkov instability sequence is a cut through a 3D simulation of the Richtmyer-Meshkov instability [12]. The simulation describes the process of two fluids mixing after a shockwave has passed through them. The images show a cross-section parallel to the direction of the shockwave. The image sequence consists of 136 images, see Figure 3. Although it seems that within this dataset changes occur only within the middle of the picture there are also significant changes to the direction of the shockwave originated.

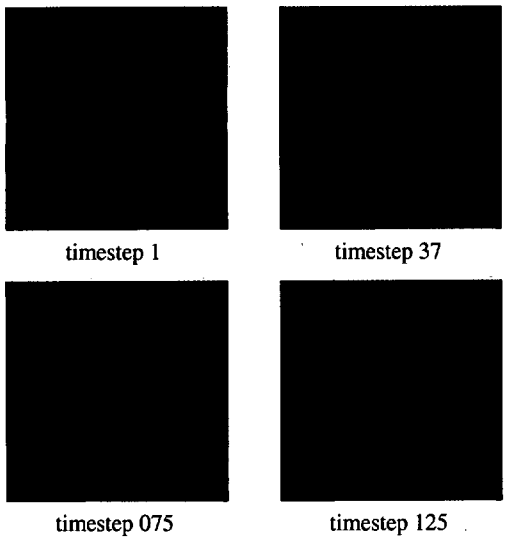


Figure 3: Images from the Richtmyer-Meshkov sequence

Strategy	Mona Lisa	Richtmyer-Meshkov
Temporal Reuse	14430	11517
Multiresolution Adaptive	17922	11686
Standard Level 3	6400	8704
Standard Level 4	25600	34816

Table 1: Comparison of number of all regions loaded over time

4.3 Examples

To evaluate our strategy we applied our adaptive approach, a time-static adaptive refinement approach and a straight forward replacement approach at the two highest resolutions to the image sequences. Table 1 shows the overall numbers of regions loaded for each strategy. It can be seen that our approach loads the smallest number of regions for the complete sequences. For a more detailed comparison, three graphs per sequence are used showing the number of regions loaded per frame, the number of regions rendered per frame, and the average pixel-error displayed per frame.

Figure 4 contains several snapshots of the Mona Lisa image sequence, with the regions replaced by our approach shown in yellow. Figure 6 shows the number of regions loaded, the number of regions rendered and the rendered error per frame for the different approaches. It can be seen that our approach loads always less than the time-static adaptive refinement. The difference between the two approaches in regions loaded per frame is an indication for the numbers of region that did not change, the number of regions rendered is nearly the same. The straight forward approach performs very poorly at lower resolutions due to increasing detail in the inner region of the image. It can be seen that our approach performs nearly as well as the time-static adaptive refinement. Both approaches show an average pixel-error of less than 1% (0,01).

Figure 5 contains several snapshots of the Richtmyer-Meshkov instability sequence with the current subdivision generated by the adaptive refinement (left column) as well as snapshots generated by our approach with the regions replaced for this frame (marked in yellow). Both approaches generate similar subdivisions, whereas our approach reloads slightly less and renders more regions especially, where pixel-error is small.

As can be seen in Figure 7, both non-static approaches have

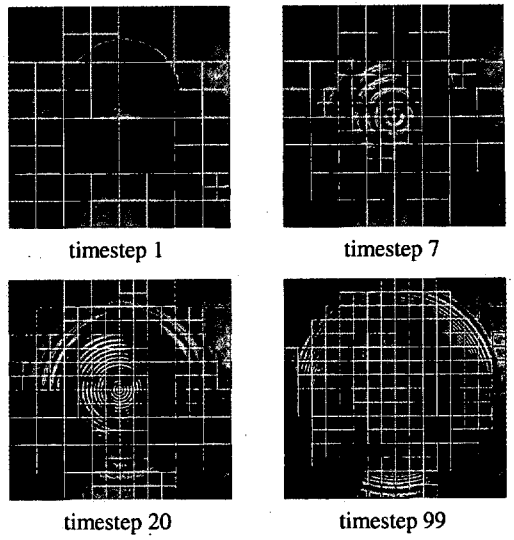


Figure 4: Images from the Mona Lisa sequence generated with our approach; regions replaced shown in yellow

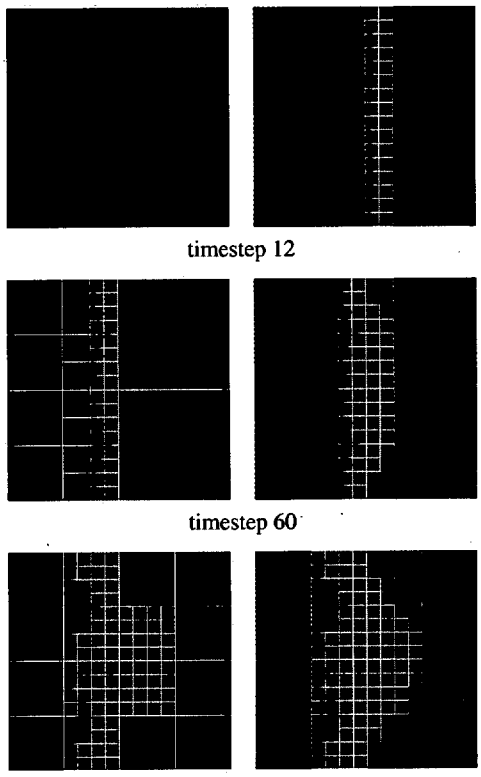


Figure 5: Images from the Richtmyer-Meshkov sequence (adaptive refinement left column, our approach right column; regions changed shown in yellow)

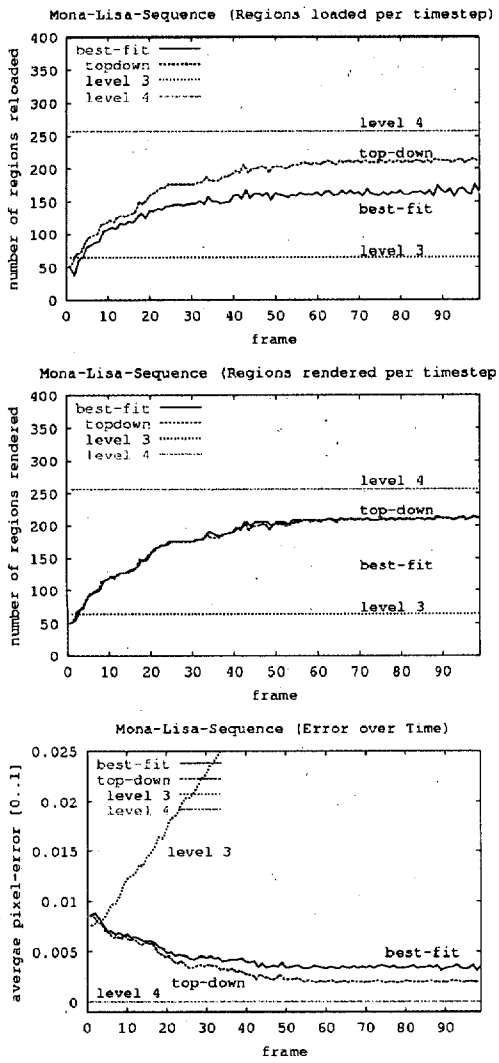


Figure 6: Number of regions loaded, number of regions rendered and rendered error per frame (Mona Lisa sequence)

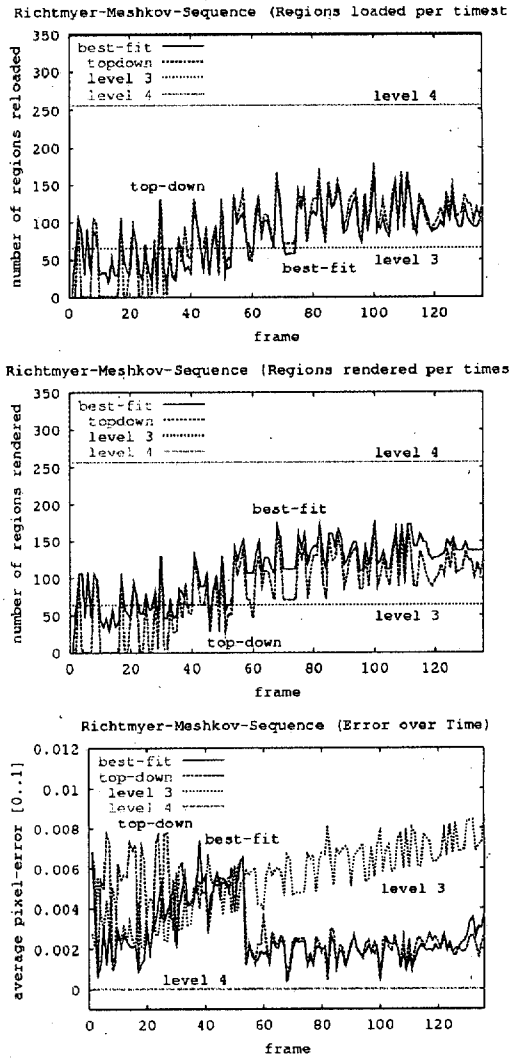


Figure 7: Number of regions loaded, number of regions rendered and rendered error per frame (Richtmyer-Meshkov sequence)

nearly the same performance with respect to the number of regions replaced. When comparing the snapshots from the sequence in Figure 5 it can be seen that our approach provides more high-resolution-regions than the static adaptive refinement method. Both non-static approaches show a high variation of activity during the first 50 timesteps. Although the outer regions of the sequence themselves seem to change smoothly over time, there is a lot of activity in the portion of the fluid where the shockwave originated. This is caused by the reflection of the wave at the contact-surface of the fluids. These changes are not visible to the eye due to the low contrast and brightness in the images but have an impact on the errors calculated at lower resolutions, forcing both approaches to continuously refine and recoarsen as the error between the different resolution changes. With the reflected shockwave traveling back, changes over time are reduced in the section of the fluids where the shockwave travels, and the replacement of regions is restricted mostly to the side where the shockwave has traveled.

4.4 Advantages

The advantage of our approach is that the number of regions replaced is remarkably reduced when compared to fixed-resolution-methods. Compared to the time-static adaptive refinement strategy our algorithm also replaces fewer regions per timestep without a noticeable increase in the visible error, providing better refinement.

The application of the errors is independent of the underlying data representation and the size of the dataset. It allows us to compare datasets of different timesteps and decide which regions to replace, without accessing the dataset itself. The size of the error tables depends only on the number of subdivisions and the number of timesteps available.

When compared to the static adaptive refinement our approach will perform well when applied to datasets where there are larger regions with high detail, but no changes over time. The high detail will force the static adaptive approach to refine and reload for every timestep.

5 Conclusions and Future Work

We have shown that our approach generates better results at a higher performance than standard replacement strategies while meeting given error-bounds and reducing the amount of data to be reloaded per frame. New methods of error calculation and approximation should be integrated in order to not only provide a mathematical sound approximation but also a visually acceptable refinement or coarsening suppressing regions to be refined where there is no visual change. In order to be able to compare two regions to determine the visual error without touching the actual data a small set of characteristic numbers should be identified that allows us to quickly determine the visual error between regions.

It is also necessary to reduce the complexity of the error tables, which currently has a complexity of $O(N^2)$ causing the size of the error tables to grow with an increasing number of timesteps.

Acknowledgements

This work was supported by the National Science Foundation under contract ACI 9624034 (CAREER Award), through the Large Scientific and Software Data Set Visualization (LSSDSV) program under contract ACI 9982251, and through the National Partnership for Advanced Computational Infrastructure (NPACI); the National Institute of Mental Health and the National Science Foundation under contract NIMH 2 P20 MH60975-06A2; the Lawrence Livermore National Laboratory under ASCI ASAP Level-2 Memorandum Agreement B347878 and under Memorandum Agreement B503159; and the Lawrence Berkeley National Laboratory. We thank the members of the Data Science Group at the Center for Applied Scientific Computing (CASC) at Lawrence Livermore National Laboratory and the Visualization and Graphics Research Group at the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis.

References

[1] A. Finkelstein, C. E. Jacobs, and D. H. Salesin, "Multiresolution video," in *Proceedings of SIGGRAPH 1996, Computer Graphics*, vol. 30, pp. 281–290, ACM Press, 1996.

[2] E. C. LaMar, K. I. Joy, and B. Hamann, "Multi-resolution techniques for interactive hardware texturing-based volume visualization," in *IEEE Visualization '99* (D. Ebert, M. Gross, and B. Hamann, eds.), pp. 355–361, IEEE, Computer Society Press, 25-29 Oct. 1999.

[3] M. Weiler, R. Westermann, C. Hansen, K. Zimmerman, and T. Ertl, "Level-of-detail volume rendering via 3d textures," in *IEEE Volume Visualization 2000* (T. Ertl, B. Hamann, and A. Varshney, eds.), pp. 7–13, IEEE, Oct. 8–13 2000.

[4] E. C. LaMar, B. Hamann, and K. I. Joy, *Efficient Error Calculation for Multiresolution Texture-based Volume Visualization*. Heidelberg, Germany: Springer-Verlag, 2002. To Be Published.

[5] K.-L. Ma, D. Smith, Y. Ming-Shih, and H.-W. Shen, "Efficient encoding and rendering of time-varying volume data," Technical Report TR-98-22, Institute for Computer Applications in Science and Engineering, June 1998.

[6] H.-W. Shen, L.-J. Chiang, and K.-L. Ma, "A fast volume rendering algorithm for time-varying fields using a time-space partitioning (tsp) tree," in *IEEE Visualization '99* (D. Ebert, M. Gross, and B. Hamann, eds.), pp. 371–378, IEEE, Computer Society Press, Oct. 25–29 1999.

[7] D. Ellsworth, L.-J. Chiang, and H.-W. Shen, "Accelerating time-varying hardware volume rendering using tsp trees and color-based error metrics," in *IEEE Volume Visualization 2000*, pp. 119–128, 155 (CP), IEEE, Oct. 8–13 2000.

[8] P. M. Sutton and C. D. Hansen, "Isosurface extraction in time-varying fields using a temporal branch-on-need tree (t-bon)," in *IEEE Visualization '99* (D. Ebert, M. Gross, and B. Hamann, eds.), pp. 147–154, IEEE, Computer Society Press, Oct. 25–29 1999.

[9] P. M. Sutton and C. D. Hansen, "Accelerated isosurface extraction in time-varying fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, pp. 97–107, Apr./June 2000.

[10] J. Wilhems and A. VanGelder, "Octrees for faster isosurface generation," *ACM Transactions on Graphics*, vol. 11, no. 3, pp. 201–227, 1992.

[11] H.-W. Shen, "Isosurface extraction in time-varying fields using a temporal hierarchical index tree," in *IEEE Visualization '98* (D. Ebert, H. Hagen, and H. Rushmeier, eds.), pp. 159–166, IEEE, Computer Society Press, Oct. 18–23 1998.

[12] A. A. Mirin, R. H. Cohen, B. C. Curtis, W. P. Dannevik, A. M. Dimits, M. A. Duchaeineau, D. E. Eliason, D. R. Schikore, S. E. Anderson, D. H. Porter, P. R. Woodward, L. J. Shieh, and S. W. White, "Very high resolution simulation of compressible turbulence on the ibm-sp system," 1999.